Kombinatorik

Christian Fuchs

1. Definition Kombinatorik
2. Grundlegende Zählmethoden
3. Binomialkoeffizienten
4. Permutationen
5. Stirling-Zahlen
6. Catalan-Zahlen
7. Zahlpartitionen
8. Aufgaben
9. Literatur

Kombinatorik

- Teilgebiet der Mathematik
- Zählen der Anzahl von Abbildungen, Elemente von Mengen, Anzahl von möglichen Anordnungen
  - z.B. Permutationen
- Wichtig für die Berechnung von Wahrscheinlichkeiten
  - Wahrscheinlichkeit nach LaPlace:

$$Pr = \frac{Anzahl\ der\ günstigen\ Ereignisse}{Anzahl\ aller\ möglichen\ Ereignisse}$$

Getrenntes Abzählen

- Zerlegung der zu zählenden Menge in disjunkte Teilmengen
- Beispiel: Anzahl aller Binärwörter der Länge n, die genau 2 mal das Teilwort „01“ enthalten
  - Aufteilen in die Menge der Wörter, die mit 0 beginnen, und die Wörter, die mit 1 beginnen

- Häufiger Fehler: Teilmengen als disjunkt angenommen, obwohl die Schnittmenge nicht leer ist

Doppeltes Abzählen

- Abzählen von nicht disjunkten Teilmengen
- Danach doppelt gezählte Möglichkeiten einmal abziehen
- Beispiel:
  - Anzahl aller möglichen Kanten in einem Graphen mit n Knoten

- Prinzip der Inklusion/Exklusion:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

Definitionen

- Permutation: Anordnung von allen n Elementen einer n-elementigen Menge
- Variation: Auswahl von Elementen einer Menge mit Berücksichtigung der Anordnung
  - Beispiel: Wieviele Möglichkeiten gibt es für die ersten drei Plätze der Bundesliga
- Kombination: Auswahl von Elementen einer Menge ohne Berücksichtigung der Anordnung
  - Beispiel: Wieviele Möglichkeiten gibt es im Lotto bei „6 aus 49“ zu tippen

Variation

- Ziehen mit Zurücklegen:

$$n^k$$

- Ziehen ohne Zurücklegen:

$$\frac{n!}{(n-k)!}$$

Kombination

- Ziehen ohne Zurücklegen:

$$\frac{n!}{k!\cdot(n-k)!} = \binom{n}{k}$$

- Ziehen mit Zurücklegen:

$$\frac{(n+k-1)!}{k!\cdot(n-1)!} = \binom{n+k-1}{k}$$

Binomialkoeffizienten

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Definition

$$\binom{n}{k} = \binom{n}{n-k}$$

Symmetrie

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n+1-i}{i}$$

Produktformel

Produktformel-Algorithmus

```
int binom(int n,int k) {
  if(k==0) return 1;
  if(2*k > n) return binom(n,n-k);
  int erg = n;
  for(int i=2;i<=k;i++) {
    erg = erg*(n+1-i);
    erg = erg/i;
  }
  return erg;
}
```

Pascal'sches Dreieck

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

Rekursionsgleichung

$$\binom{n}{0} = 1, \quad \binom{n}{n} = 1$$

Basisfälle

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Rekursionsgleichung

Algorithmus für Binomialkoeffizienten

```
long long koeff[N][N];

void binom() {
    koeff[0][0]=1;
    for(int i=1;i<N;i++) {
    koeff[i][0]=1; koeff[i][i]=1;
      for(int j=0;j<i;j++) {
        koeff[i][j]==koeff[i-1][j]+koeff[i-1][j-1];
      }
   }
}
```

Zeitkomplexität

- Produktformel: $O(min(k, n-k))$
- Rekursiver Algorithmus: $O(n^2)$
- Achtung: Werte können sehr groß werden, deshalb long long oder BigInteger verwenden
  - Abschätzung des mittleren Binomialkoeffizienten

$$\binom{2n}{n} \in O\left(\frac{4^n}{\sqrt{n}}\right)$$

Permutation

- Mathematisch: Eine bijektive Abbildung einer Menge auf sich selbst
- Schreibweisen:
  - Explizites Ausschreiben: z.B. acdbe
  - Zykelschreibweise: (a)(bdc)(e)
  - Matrixschreibweise: $\begin{pmatrix} a & b & c & d & e \\ a & c & d & b & e \end{pmatrix}$
- Für eine n-elementige Menge gibt es n! Permutationen
- Im Wettbewerb zu hoher Aufwand alle Permutationen zu berechnen
- Im ICPC wird meist nur nach einer bestimmte Permutation gefragt, deren Position in der Menge der geordneten Permutationen bekannt ist

```
int * computePermutation(int *a, int n, int m,int *res) {
  int done[n];
  long fac = faculty(n-1);
  for(int i=0;i<n;i++) {
    int pos = 0; while(done[pos]) pos++;
    while(m>fac) {
      pos++;
      while(done[pos]==1) pos++;
      m -= fac;
    }
    res[i]=a[pos];    done[pos]=1;
    if (i < n-1) fac = fac/(n-(i+1));
  }
  return res;
}
```

Permutationen in lexikografischer Ordnung

```
void getNext() {
  int i = N-1;
  while(Value[i-1] >= Value[i]) i--;
  int j = N;
  while(Value[j-1] <= Value[i-1]) j--;
  swap(i-1,j-1);
  i++;
  j = N;
  while(i<j) {
    swap(i-1,j-1);
    i++;j--;
  }
}
```

Probleme bei der Permutationsberechnung

- Mississippi-Problem:
  - Wieviele Anagramme von „Mississippi“ gibt es
  - Problem: Buchstaben kommen mehrfach vor
  - => Von den n! Permutationen sind mehrere gleich
  - Lösung:

$$\frac{11!}{4! \cdot 4! \cdot 2!}$$

Zeitkomplexität

- computePermutation: $O(n^2)$
- getNext: $O(n)$
- Laufzeit jeweils für genau eine Permutation
- Für alle Permutationen $O(n! \cdot n^2)$ $bzw.$ $O(n! \cdot n)$

Stirlingzahlen 1.Art

- Anzahl der Permutationen einer n-elementigen Menge mit genau k disjunkten Zykeln
- Schreibweise: $s_{n,k} = \begin{bmatrix} n \\ k \end{bmatrix}$
- Beispiel:

  Menge {a,b,c,d}

  (ab)(cd),(ad)(bc),(ac)(bd),(a)(bcd),(a)(bdc),(b)(acd),(b)(adc), (c)(abd),(c)(adb),(d)(abc),(d)(acb)

  => $s_{4,2} = 11$

Rekursionsgleichung

$$s_{0,0} = 1 = s_{n,n}$$
$$s_{n,0} = 0 = s_{0,k}$$

Basisfälle

$$s_{n,k} = (n-1)s_{n-1,k} + s_{n-1,k-1}$$

Rekursionsgleichung

Algorithmus Stirling 1

```
long long koeff[N][N];

void stirling1() {
    koeff[0][0]=1;
    for(int i=1;i<N;i++) {
      koeff[i][0]=0; koeff[i][i]=1;
      for(int j=0;j<i;j++) {
        koeff[i][j]=koeff[i-1][j-1] + (n-1)*koeff[i-1][j];
        }
    }
}
```

Stirlingzahlen 2.Art

- Anzahl von Zerlegungen einer n-elementigen Menge in k nichtleere, disjunkte Teilmengen
- Schreibweise: $S_{n,k} = \left\{ {n \atop k} \right\}$
- Beispiel:

  Menge: {a,b,c,d}

  {{a,d},{b,c}},{{a,b}{c,d}},{{a,c},{b,d},{a}{b,c,b},{b}{a,c,d},

  {c}{a,b,d},{d}{a,b,c}}

  => $S_{4,2} = 7$

Rekursionsgleichung

$$S_{0,0} = 1 = S_{n,n}$$
$$S_{n,0} = 0 = S_{0,k}$$

Basisfälle

$$S_{n,k} = k \cdot S_{n-1,k} + S_{n-1,k-1}$$

Rekursionsgleichung

Algorithmus Stirling 2

```
long long koeff[N][N];

void stirling2() {
    koeff[0][0]=1;
    for(int i=1;i<N;i++) {
      koeff[i][0]=0; koeff[i][i]=1;
      for(int j=0;j<i;j++) {
        koeff[i][j]=koeff[i-1][j-1] + j*koeff[i-1][j];
        }
    }
}
```

Catalan-Zahlen

$$C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$$

$$C_{n+1} = \sum_{k=0}^{n} C_k \cdot C_{n-k}$$

- Die Anzahl der Möglichkeiten ein konvexes (n+2)-Eck durch Diagonalen in Dreiecke zu zerlegen, wird durch die n-te Catalanzahl beschrieben

Interpretation der Catalan-Zahlen

- Die n-te Catalanzahl ist die Anzahl an Möglichkeiten in einem Produkt mit n+1 Faktoren Klammern so zu setzen, dass jeweils nur zwei Faktoren multipliziert werden
  - Beispiel: $C_2 = 2$

$$Klammerungen\ von\ x_1 \cdot x_2 \cdot x_3 :$$
$$(x_1 \cdot x_2) \cdot x_3$$
$$x_1 \cdot (x_2 \cdot x_3)$$

Interpretation der Catalan-Zahlen

- Anzahl der Random-Walks von 0 nach 2n, so dass sich der Pfad nie unterhalb des Startpunkts befindet und bei 2n auf gleicher Höhe mit dem Startpunkt liegt
- Die n-te Catalan-Zahl bezeichnet die Anzahl an möglichen Binärbäumen mit n+1 Blättern

Catalan Algorithmus

```
long long koeff[N];

void catalan() {
  koeff[0]=1;
  for(int i=1;i<N;i++) {
    for(int j=0;j<i;j++) {
      koeff[i] += koeff[j]*koeff[i-j];
    }
  }
}
```

Laufzeitanalyse

- Asymptotische Laufzeit zur Berechnung der ersten n Catalan-Zahlen:

$$O(n^2)$$

- Summe der ersten n ganzen Zahlen:

$$\sum_{k=1}^{n} k = \frac{n \cdot (n-1)}{2} = \frac{n^2 - n}{2}$$

Integer Partitionen

- Integer Partitionen dienen dazu die Anzahl an Möglichkeiten eine Ganzzahl n in k ganzzahlige Teilsummen aufzuteilen zu bestimmen
- Es gibt 2 Arten von Integer Partitionen:
  - Geordnete Partition
    - z.B. n Euro auf k Leute verteilen
  - Ungeordnete Partition

Geordnete Partition

- Es wird die Reihenfolge der Teilbeträge unterschieden
  - Für n=4 und k=2 gibt es die Lösungen:
    - 1+3
    - 3+1
    - 2+2

- Abbildung auf Kombination mit Zurücklegen:

$$\binom{n-1}{k-1}$$

Ungeordnete Partition

- Reihenfolge der Teilbeträge wird nicht beachtet
  - Beispiel: n=4 und k=2
    - 1+3
    - 2+2
- Rekursiver Ansatz:

$$P_{0,0} = 1 \qquad P_{n,0} = 0 \quad \textit{für } n \geqslant 1$$

$$P_{n+k,k} = \sum_{i=1}^{k} P_{n,i}$$

Berechnung der ungeordneten Partitionen

```
long long koeff[N][N];

void unorderedPartition() {
  koeff[0][0]=1;
  for(int n=1;n<N;n++) {
    for(int i=0;i<=n;i++) {
      for(int j=1;j<=i;j++) {
        koeff[n][i]+=koeff[n-i][j];
      }
    }
  }
}
```

Laufzeitanalyse

- Geordnete Zahlpartition: $O(min(k, n-k))$
  - Berechnung eines Binomialkoeffizienten

- Ungeordnete Zahlpartition: $O(n^3)$

Aufgaben

10784 Diagonalen

Für jedes N gibt es ein konvexes n-Eck, das mindestens N Diagonalen hat. Für ein gegebenes N soll nun das kleinstmögliche n-Eck gefunden werden, so dass jedes m-Eck mit m < n weniger als N Diagonalen hat.

Eingabe: Anzahl N an Diagonalen

Ausgabe: kleinstmögliches n

11027 Palindrome Permutation

Für einen gegebenen String gibt es Permutationen, die Palindrome darstellen, also rückwärts gelesen das selbe ergibt wie vorwärts gelesen. Diese Permutationen kann man wiederum lexikographisch anordnen.

Eingabe: Zu permutierender String s und eine Zahl n

Ausgabe: Die n-te Permutation von s, die ein Palindrom ist

357 Let me count the ways

Auf wieviele mögliche Arten kann man sein Wechselgeld zurückbekommen, wenn es folgende Münzen gibt:

- Penny: 1¢
- Nickel: 5¢
- Dime: 10¢
- Quarter: 25¢
- Half-Dollar: 50¢

Eingabe: Wert des Wechselgelds in ¢

Ausgabe: Anzahl an Möglichkeiten

Literatur

- http://de.wikipedia.org/wiki/Kombinatorik
- E.W. Dijkstra
  A Discipline of Programming
  Prentice-Hall 1997
- Steven S. Skiena
  The Algorithm Design Manual
  Springer 2008

Fragen?

Vielen Dank für ihre Aufmerksamkeit